From frank@funcom.com Tue Nov 9 13:23:29 1999
Date: Thu, 28 Oct 1999 12:57:37 +0200 (CEST)
From: Frank Andrew Stevenson <frank@funcom.com>
To: livid-dev@livid.on.openprojects.net
Subject: [Livid-dev] Working PlayerKey cracker

In response to feedback from yesterdays post I have now refined
my attack in the following ways:

The CSSdecrypt key can now be recoverd with only 5 bytes of
known output. Sometimes multiple keys will be found to a
single output, due to colissions in the mixing stage. But
this is not a problem when recovering KEKs (Key encryption
Keys), as all keys found will be equivalent / interchangable.

There has been some debate around the 'hash function'. I choose
to view it as a very simple encryption function. With 5 byte
input, 5 byte output and 5 byte key. When searching for a
player key, the input / output is known. The cipher can then be
attacked with a complexity of 2^8. Code for the key recovery
is given below. This cipher has many colissions, and some
input outup pairs have no keys, while others have multiple.
The latter is a concern when searching for Player keys, as
they have to be eliminted by checking agains other discs.

I have attached a program that works as follows:

```
hippopotamus:~/tmp> time ./keyrec 22 e1 67 83 72 0f c1 7a 96 98
Recovering Key
Possible mangling key: af c9 07 42 1f
  Possible Player key 51 67 67 c5 e0
  Possible Player key 69 d2 e3 92 ae
5.000u 0.010s 0:05.44 92.0%        0+0k 0+0io 87pf+0w
```

Here 2 equivalent player keys are recovered from the
```
input:  22 e1 67 83 72   - Disc key
output: 0f c1 7a 96 98   - intermediate key, common for all player keys
```

The process takes 5.5 seconds on a PPro200, somewhat slower
now that only 5 bytes are known in the keystream.

If this works, as I hope it will, I will leave it as an exersice
to the reader to recover all player keys :-)

 frank

```
-------------- This is how to recover the 'hashing key' --------
static int unmangle ( unsigned char* in , unsigned char *out ) {
  unsigned char A[5];
  unsigned char B[5];
  unsigned char C[5];
  unsigned char k[6];
  int i,j;

  /* Recover mangling key */
  memcpy( A, in, 5 );
  memcpy( C, out, 5 );
  k[5] = 0;
  for( i=0 ; i < 256 ; i++ ) {
    k[4] = i;
    for( j = 4 ; j >= 2 ; j-- ) {
      B[j] = k[j] ^ CSStabl[ A[j] ] ^ A[j-1];
      B[j-1] = CSStabl[ B[j] ] ^ k[j] ^ C[j];
      k[j-1] = A[j-2] ^ CSStabl[ A[j-1] ] ^ B[j-1];
    }
    B[0] = CSStabl[ B[1] ] ^ k[1] ^ C[1];
    k[0] = B[0] ^ CSStabl[ A[0] ] ^ B[4];

    if( ( CSStabl[ B[0] ] ^ k[0]   ) == C[0] ) {
      printf( "Possible mangling key: %02x %02x %02x %02x %02x\n", k[0],
k[1], k[2], k[3], k[4]    ;
    }
  }
  return 0;
}
```

------------ The following is the complete sourec for ------
----------------- player key cracker ------------------------

```
begin 640 keyrec.c.Z
M'YV0(J*X&<.F#IDR('C,H4,FS1L7:'PT4!!P8,&#"1?*$7@&HD2*`@D:1,AC
M#)T\<,IXG%A1)$:%#-FD$;-2P<2%8>BD&0.BCILY:<ZX*4,&A$`Z((9,F4(G
MC!@86V+$Z-)C3PT6,%C$8"&#Q0P6-+!JY>H5;)\=$V\VU<G3)U"A1$&,01-&
M3M*E3<7$V"*CA@VJ$_=,3``#SXROA6\@-BP&*QX9-AS;6"QCL8W&B2,7GE'&
M\8W.FS4_QHS'!NC'IVV(OM%X<&$RBW,L)D,:AF@:BV$LID$ZA^B#CG.<)B,:
M!FD:IV&<IB$Z1VL%A/'4N.$X!O7"-<PXQM&U\)CK>'"`'Z.]<(SNTL-45X\=
M/8[R>,:P#S]_#/H8VEWC(9,C>/_76Q4&0U:%T?`?'C`<2$.`>.1`X'XX!!?A
M:P_"P"`-$R*8(0T/YK"5?C.0X9F(FUU5F`QA%68#B8^Q:(.)>-R0HF%C>%;C
M9C7V)=F-C_%HPXPW7*4?&2SFP"(9,,(P(PTLPL`B#3#F,",9/.;`(QDSP@`C
M#3S"P",-,^8@)'38'1C#@34PB,.#8QR(PX%C,!C#@S5D&$.&-3R(`X-C9(A#
MAF,\*!4+0X*7`WADP`<#>C2`!P-X-,"7`WIDS)?#?&2@!P-\-,P'PWPTH)=#
M?F1*MU@,B]5`&@ZBC;$8#HN-05H,HM5P6@RG^;4=:6.<AL-I8X@6PW/1B;&8
M&8N)05H8HI6Q6!B+E4&:&:*)<9H9IXDAFE..E7%:&*>5(9H9Q&X&W@W@SO"?
M#.C9`)X,X-D`WPWHS3#?#?/-@)X,\-DPGPSSV8#>#:06>Z`9!^KE6!@/EG%@
M##/6P&,/-8P(PXPCL$C#CR.,6,,8Q;+HADLB@%C&#,2M3"+9<!HQHB\&@&
MCV+,&`:,9?`8!H]ES&A&T9L=>,.!,S`HPX,V'"C#@38P>,.#,V1X0X8S/"@#
M@S9D*$.&-CQX`\G%@F<&>&+`%P9Z98`7!GAEP&<&>F+,9\9\8J`7!GQES!?&
M?&6@9X89$YV5E@(XL=733T$-5=1<==W%E%,R\.478`H(5NJ`CEG8NZ:Y99ED
M<8X>"$.&QO7N:9-=*MJ9?N=5=ZJ@<@H+'LW5S3SK?&=69R=\N%9W<HWZH>B8
MCB>*!N_Y>JN]+V4LRL`CO^>?]O;Y?I,&,*&ESB#:#.F:TOQ@I"_'',:`=&.0
MN@QXFA`9D$<S((V]#(BV".G'0([!4`:/`ZHG?8E3RWG0@C+(J-TL:4NWH8Y^
M5.68&LRG!FC"$WQLU<*4Y0P]-4@5G1BD*^R`1V<M)%MI7.2C?J4&;H`CC;\D
MN,),*CBQZ1"#?!,<0X4I2KVQU(%RD*%1!><T1@H.^4JUINVHR3VOVMG*1".>
M[;C)3ZN:#\RV\S/X_(I_Q:H6XJ@&H\PYQEB?K!B#SO?);"4-:Z2IW"<39L%2
M,6UA2]N6XQX4!@8M;F'78M'7%J:X;SW,8\M2SY!(4RG'\,>8&4J4,8=SI"M1
M6^\\FM6F9;F#76Q<AA.;U$1W+)&U$0].*TS7NL6Y<$4,9*3I7+>:QK5F0:YA
M'&/<#$J'%IN@;BT[6=U;7"<7NMA%*;,SUA;2)`/<Z2XZO#O1DN!6),K]9EJ*
M*DZ%>KZF#M&G-I5J8=RIU*'B`8-.Q2I4;[K5F%:5IS05*QZBBM2I=O6L1$VK
M4;/:UK)25:5H#>M<QZI5/'!U=UX%ZU7WNE:R^M6L>(VK7AMTU/TD%;!P_6I1
M&4M7QTY5IY$5[(K4RE;+VO6MB96L7"G+U[H>]]JY,>M@2500X5I8&=+!M+
MZ]G38C:TFGUJ:TW[VMNF5K&KG>UN:^M.WYH/N)LE;&=IXU;(XE:VC66N78UK
M^)YWL>YU;4[M^UO1XC>]E^7O<?T;7T_!_UKG]S2UGWXO8!''.7MMZ%[U(('K&#E
M5\4FCG&&9[SA&H?WO_0M+XLK?-5T>C=X5T=8BEE8V2%HJ/J-_C_&>>9MAW[<X""+V,8I3C)QERQ?(-,X
M62IOR?KFDLL9]A+;?V=G[SF-4\]9SX6VRX^K+.@T'_C-9V[T6TSHF%,8C!S6<>5
M+G*8Z:SG8X^.HJ=+74;=TKJXP.ZC>'$*#6SWEQXS)L:8+!";]$9+MC5,4!VB
M4S3I5C'!B2PK2O73A@#E-X\!#F1VZB.>DHFWC-GM8JU$Z<Z6U+(<::V)31M:
MF&HV)O82#4AB"XM2?_:EH,W/:SX"V\VJE+,K01!K7K3:(8'J'D:9<U^::Q?C
M[B,J'8*J7IO;F.BZHE1)@JH^__)<PS#'/4N]$8/YPMR]+"?,W3'R-JU2W[:@
MV"QA-:=6O_E?M58SKL@HU5?-_9;]OI6:<-T*D+D:3@.SI<5K/6]W:KHOG]1F
M2[YQ3$YSY.$T83A*N4WLGPERI&JS>:*'P2UB9MHCF@YTS+*MTC,'ZX]25[:E
MGL$(?7C8FF2X-K,HP4AEQB1@)\\H-2$Z:3M/BEG\=.DBJ)VL2"D[$HL<"84N
M'DU$2GW5;F)8F6=9)EJGBDVJ9K.8`T)5C<>RXNY6=1Q>Z6]92DSHK'I#&A::
M?)*:(7R4N1GXD#(QDB.=`DQG45&WA75P>9U'99>7&K#:I/J9,Q:A"I\8OD;X
MP0$^0273^_/M/H.Y[]U^#^<9VAM0]BV$?75<+QG6GT_U&3R]HVY/>L^D4H+$
MU'F<5QV:)QF8=SZ6MT&[TCNEARQI]$F-QW@MI'C2TRV'ATOG0W@9)'B](WG1
M@3OC\DE[IW<U8!QWURUU]S3G(W<9]#)-`GCMQ':?I'9I8T3!47;=`B-.5!AB
M`73=XG,;<SX[ET$YUSO_!$^'07,,M$PM!'/5X7*2P7+UHTTIMQW)<7,?2'*5
MY3_&!'+!X7'=PG$+HW':A'';41PFUTX3]TD19TP/8QP-URT+MS`)ITT'MQV>
M4G$?*'"?!'#&Y&]J83T"I6\-@V_:9&_;H2D$UT[Q]DGO9DSM)DC5H6X[LC#G
MIDWE1DF],V];!6Z?Y&W&Q&W!H6T+Q4O+8VWHE$'3UCOBUD[/]DG-ADPMI&S>
M80=OD`98`5(',0=C(`=AT`9BP`9E@`(:56K28SLJL`9ED`<@D`*O!AVD8GU&
M X09(00=;00==00=?00=A00=700<V48$)T)`<=11&45&#(9$]``(&F0=;``-=
M``)>``+F,2`>29$GF9)1T04S.0,]@`(J@`(H`))Q(9(JD`(HD)(K(`,ID)0S
M20,]8)$F<`,YV9OSH`(RL`(XT`(8.9,UT`,P48$@\)5@^95PL!$0:08`*O)+
M<)"I@Q`Y`O(X(O=TH`,@4`*Z@O=SN2AV29<X@`=<X`8BP`(@()&`29&#.O,)
ML`().1@GEO`8VO,@E1>UOY?=X`6MZ1OSL`42B9.#H9HRT)OQX`,^$`,>J9HQ
MFO*^B9S.^9V^R9SE*9X^4`,I8`)E89P6N9,6J9S,RO<<"9_9*9VKQI'6FO"J
MT)>4&:&E<SJG\P(J4*(F>J(HFJ(JNJ(LVJ(J,!$@\*(*$*-?2O5H@!!$D`9V
O(]LZO:O\P9N$`=U4`9VD1--,O9K`*,R2J,@O`1EL*.]4A1AX*,@<`<;OO=T
MT`8N8*=M<)HSJ@)?*O5EX(]A,`=Q`:O@$`8@<`1OP`O@(!.]\A-EP*9]ZJ*4
M6JF6BJ(O$&KP^)!((O4^PO1F,`=R(`-"\#5K<`=UOO9SH)DO"O)V,#MP.3M9
MNA9EX)4@<)F9^9DG"O.CFO:@::M9BJO;6IJZRJM@*9$@<)(6"O(F`).'1:U?
M*@>/&3F8V2MNV3ER,!?5BIE<F@=9ZI9HD`9FD*6_NJ4JD*GX>K'7J@:<Z9EJ
M`)I]8O.CJO;BF@*[6JQA":\@4)R&N:[F,3JC([-\<+,@L++*&9HW<*]A^:[9
MN9U*VYT^\)WI.9[@:9X^@)[AZ9OL^:"CXZY@Z;'RZK(]@*XQH*MB(`>=LP:>
M"O(T"P(8NJQK^Y5D2P=U(`=;:I$5M;9JD1.C!GT<=6H/:O=YL6HM:YUWJSKR
M2(^RDX^F2JO`Z:L=51=^"@=,O;;[")B,R[<J``=4X!3^&`24J[>F-GV7F[G]
M*I)IH*SYZJ=44`8'2P=T@128J[EE$`1&D:N(FA)RT`:/B;>&.K+8V@9ET`:$
M2@>LZK>J!IA9`O(M:[OFJYGG"I/;RK)^$:ZF*;,"(;Y9LP4@T+RC&P1;D`8M
M:KT89:A$*@>8Z:'H^Y5E@`=I`+X@T`)A"Z)L>SK8^Y`GG`9A(!-Z@!#*BQ#S
M2Z2$"@)O8`:!Z;PT$*,D^Y4#C*WLJ\#NZ[(*S,`/K&KVB[^;N[_]N[[#Z\+I
R\8ZVZYSW*PPJ;4:N<7$"Y;Z>L)$FA,(,09I``<W:A>&J17<FZO_"@)D:Z<
O`9V,*42B\F:'+8)6[TSC)"D?+2!R932^9HE)9NI)E*WJ;7ONIL2.9[(')C#
Z9/&B9ST"9XTT,RK"<6D2YW6Z<OZ>A!D4`<2RY'J&I@U(,1IB[3K"[F4W#EM
$,$@@)-A><X-+*_E++/B'*^!Z;*F658)^:";8@;-S+7Y#)H8J:OB'*#L.B=R
K:_8$A<O<*BR3,LV;)@MW<>!?+0VB[,)<<#,"O(_2ZP,W<<46J]::[<EJZ]'
,*1SX,@W"@)M\`8'NU$>NQ-3BA1B<,(\[,.VO,DH'9:[V[OT2,CK"P*>Z@:@
*JJD:JJH*@>J*J^&"9B&R<)A*=``V=4^^M4;>\?'HZM?R[XRB\;E:IA\<)+L
:BS#4M@+[<MT[;M@+=AC7=:C6JI(FM9KS:P@X-8CK+9I*]>NRKMU_;L(@=?V
*K-[K:U]+<\>S9A&$`9IX(]÷\<AG@-2Y6M5(T0+[:L(ZX09G(,)P?;0E?,(`
J<*73=1A&<A9JH\;R\E-3;:!F0:-')@`.\LRX=ERX=72G9);#99%G,H@4,ZL
;,&^')W÷+,RQ*9VU><P1K<R)^9O6_,S÷&0/'F9S+2<W6S)38+`3:;-':N;0S
X)U1JY[D6;53V]]6VY[O&=$Z3;/3;)^O^]/9B<WZRY_6#*`"2J`0S=!RW<=W
```

```
MW+,@0,'W&[GL+,5B*[EF&]$3Z@,5.M16_)4"?<HG6<^NS,U^FI8(B:JY>ID^
M412TK:$\_)BOC+QUP`9(4=,U&\3XG-AVC1`MS<DRF^$\BS'SS=/.RMF*?=?L
M"I7RW-%DN:%G*>5'+I=T.0-YB9=WN9<>2KYXH-!\@`)>?)XIP`)>[,_NM-G4
M#=8G+>1@>=J4"0)0X-1`,;IZS@9AD`=$BI('>9<+:NAA?N@BC>AE3L:V>YZ`
M^>8"$N<8B;4^#)C%&<1LKJYV]<MM[`.=?)@!');"C=D61=03H:]`T09PX(_5
M6J<Z8:B/+,EV0*1(<:-VV:59ZL@`*Y0DJZ\)FZ5R>0>KS08@<`8ZBA!@2@9W
MVN-S\.-TD*L>B\5`BNL@T!#('NV'JMP"$9E!3K*#FU%\B^O8+LI1<0/^RY@6
M4AE0XBYOXA0N$SE<``,BH*S.9ZQ(4,)"H+%E0`5OD`00"9"2?+"6"P=.4`?\
M..BEO:Q\.Q?!VZIR\,<6BZYOP`9O<`>*>[\'G_!RL)(MN=MV\;\NZZ<H<!1$
MZ9:4/!?72NY!(<J`:;$R2]LLG^VYNML3[]P6C_'7:O`(;RT=/Q7H*P<(7?(0
M>?(:H?(`.?,NWU&Z*O,E7.[:SL+%^K9QNZ5V@==9^\+S.\.&V^T\3NQL8.QT
M4>OW"\+?++%G4!>5<P8(0<I]$,`CZJ>`6NMPR=13>@8R@=NO/A"QOJ6TW;MR
MH8](2NAY,`>D?.\BZ1-M</>NOKBQV[ARX*<"4=D!Z9"6ZYA(L?"]^OA\J[\U
ML-&5.WWV_?FXR_G3-POC!?JF'SMKL`5_\?!(D0;(^L?Z"JAV2O=VC]MY?P:$
M?]W<VP9C``=Y`)"<^Y"`6<_4^OO!/_Q)`9B8?_SHV_J?[\=6;,HI^YD'?+[=
MK?E@V?K52<5:6\0K>Y)!K+;M?)*<K+(MO-O<'Y;VKO;IWOKP[Y+4YQ1[,:9;
M,/\M^9+ZJP8JO-'N+__]O_4EG>[?^YMB+TG^[;^DD/\`8/<3@$#O)/6_%E!2
MZ%\!M%\3$.@MP/?W_U!<R;)O+(D`%C,#>)/>&?UK?4#O):&^J:"U6E\(/$D@
M<`%>0/P7`C?@%JA.?TS%+2:`-`-CH`E,@![O*Z6`O8;Z0F#[NW-9+H/QN9G@
MZA:?[N,(A,_+B;E%1P6G(%YJ="ZP"P`F%*@%41)?Z(*M;P:`P1OX\9H92$%<
M2"KCKO'`I)_88#$+8KLMGI6LMF7JIEX9@%MR:X^%*(N2ZOQ4$U!M6XI.\3V'
M(+)(EDA:?`*!52&%NG`&:@3D\U.,T`YL@8_WXHQ5%5M6H2]V"(34M[[V`$P2
M"P0"*ZBMTK=1/E?LP'R<\"1YPE#("D>A%1-HC)`G;+C_M<(JH5A*@F>I"LR!
M,,#VO%RNP@-V"1B"`&%(#.U2'D!(QQ`$),-EB)``4H["1[VOI16!@6"[X`#(
M:E)I8`Z(*>N6`AI=)`R"<;#'X4&_!YR*5:DS97X,^VTW^.7=MI8;<&?I+M_A
M@7V7I?P=P$-AWW"!`;UPB`H_4SS4=_S.'@:\;7<&).$"^POHJ]2I.HIEL:Q5
G[ZMI>,[VO0&Z!P5EG&XK?4X0[_TCX[?C,A^ULH/C$""1MKHEHBP*
`
end
```

This sentence is unique in this respect; it can safely
be attributed to my employer, Funcom Oslo AS.
E3D2BCADBEF8C82F A5891D2B6730EA1B PGPmail preferred, finger for key
There is no place like N59 50.558' E010 50.870'. (WGS84)

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>

static unsigned int CSStab0[11]={5,0,1,2,3,4,0,1,2,3,4};

static unsigned char CSStab1[256]=
{
        0x33,0x73,0x3b,0x26,0x63,0x23,0x6b,0x76,0x3e,0x7e,0x36,0x2b,0x6e,0x2e,0x66
        0xd3,0x93,0xdb,0x06,0x43,0x03,0x4b,0x96,0xde,0x9e,0xd6,0x0b,0x4e,0x0e,0x46
        0x57,0x17,0x5f,0x82,0xc7,0x87,0xcf,0x12,0x5a,0x1a,0x52,0x8f,0xca,0x8a,0xc2
        0xd9,0x99,0xd1,0x00,0x49,0x09,0x41,0x90,0xd8,0x98,0xd0,0x01,0x48,0x08,0x40
        0x3d,0x7d,0x35,0x24,0x6d,0x2d,0x65,0x74,0x3c,0x7c,0x34,0x25,0x6c,0x2c,0x64
        0xdd,0x9d,0xd5,0x04,0x4d,0x0d,0x45,0x94,0xdc,0x9c,0xd4,0x05,0x4c,0x0c,0x44
        0x59,0x19,0x51,0x80,0xc9,0x89,0xc1,0x10,0x58,0x18,0x50,0x81,0xc8,0x88,0xc0
        0xd7,0x97,0xdf,0x02,0x47,0x07,0x4f,0x92,0xda,0x9a,0xd2,0x0f,0x4a,0x0a,0x42
        0x53,0x13,0x5b,0x86,0xc3,0x83,0xcb,0x16,0x5e,0x1e,0x56,0x8b,0xce,0x8e,0xc6
        0xb3,0xf3,0xbb,0xa6,0xe3,0xa3,0xeb,0xf6,0xbe,0xfe,0xb6,0xab,0xee,0xae,0xe6
        0x37,0x77,0x3f,0x22,0x67,0x27,0x6f,0x72,0x3a,0x7a,0x32,0x2f,0x6a,0x2a,0x62
        0xb9,0xf9,0xb1,0xa0,0xe9,0xa9,0xe1,0xf0,0xb8,0xf8,0xb0,0xa1,0xe8,0xa8,0xe0
        0x3d,0x1d,0x55,0x84,0xcd,0x8d,0xc5,0x14,0x5c,0x1c,0x54,0x85,0xcc,0x8c,0xc4
        0xbd,0xfd,0xb5,0xa4,0xed,0xad,0xe5,0xf4,0xbc,0xfc,0xb4,0xa5,0xec,0xac,0xe4
        0x39,0x79,0x31,0x20,0x69,0x29,0x61,0x70,0x38,0x78,0x30,0x21,0x68,0x28,0x60
        0xb7,0xf7,0xbf,0xa2,0xe7,0xa7,0xef,0xf2,0xba,0xfa,0xb2,0xaf,0xea,0xaa,0xe2
};

static unsigned char CSStab2[256]=
{
        0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x09,0x08,0x0b,0x0a,0x0d,0x0c,0x0f
        0x12,0x13,0x10,0x11,0x16,0x17,0x14,0x15,0x1b,0x1a,0x19,0x18,0x1f,0x1e,0x1d
        0x24,0x25,0x26,0x27,0x20,0x21,0x22,0x23,0x2d,0x2c,0x2f,0x2e,0x29,0x28,0x2b
        0x36,0x37,0x34,0x35,0x32,0x33,0x30,0x31,0x3f,0x3e,0x3d,0x3c,0x3b,0x3a,0x39
        0x49,0x48,0x4b,0x4a,0x4d,0x4c,0x4f,0x4e,0x40,0x41,0x42,0x43,0x44,0x45,0x46
        0x5b,0x5a,0x59,0x58,0x5f,0x5e,0x5d,0x5c,0x52,0x53,0x50,0x51,0x56,0x57,0x54
        0x6d,0x6c,0x6f,0x6e,0x69,0x68,0x6b,0x6a,0x64,0x65,0x66,0x67,0x60,0x61,0x62
        0x7f,0x7e,0x7d,0x7c,0x7b,0x7a,0x79,0x78,0x76,0x77,0x74,0x75,0x72,0x73,0x70
        0x92,0x93,0x90,0x91,0x96,0x97,0x94,0x95,0x9b,0x9a,0x99,0x98,0x9f,0x9e,0x9d
        0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x89,0x88,0x8b,0x8a,0x8d,0x8c,0x8f
        0xb6,0xb7,0xb4,0xb5,0xb2,0xb3,0xb0,0xb1,0xbf,0xbe,0xbd,0xbc,0xbb,0xba,0xb9
        0xa4,0xa5,0xa6,0xa7,0xa0,0xa1,0xa2,0xa3,0xad,0xac,0xaf,0xae,0xa9,0xa8,0xab
        0xdb,0xda,0xd9,0xd8,0xdf,0xde,0xdd,0xdc,0xd2,0xd3,0xd0,0xd1,0xd6,0xd7,0xd4
        0xc9,0xc8,0xcb,0xca,0xcd,0xcc,0xcf,0xce,0xc0,0xc1,0xc2,0xc3,0xc4,0xc5,0xc6
        0xff,0xfe,0xfd,0xfc,0xfb,0xfa,0xf9,0xf8,0xf6,0xf7,0xf4,0xf5,0xf2,0xf3,0xf0
        0xed,0xec,0xef,0xee,0xe9,0xe8,0xeb,0xea,0xe4,0xe5,0xe6,0xe7,0xe0,0xe1,0xe2
};

static unsigned char CSStab3[512]=
{
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6c,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
        0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb,0xff,0x00,0x24,0x49,0x6d,0x92,0xb6,0xdb
};

static unsigned char CSStab4[256]=
{
        0x00,0x80,0x40,0xc0,0x20,0xa0,0x60,0xe0,0x10,0x90,0x50,0xd0,0x30,0xb0,0x70
        0x08,0x88,0x48,0xc8,0x28,0xa8,0x68,0xe8,0x18,0x98,0x58,0xd8,0x38,0xb8,0x78
        0x04,0x84,0x44,0xc4,0x24,0xa4,0x64,0xe4,0x14,0x94,0x54,0xd4,0x34,0xb4,0x74
        0x0c,0x8c,0x4c,0xcc,0x2c,0xac,0x6c,0xec,0x1c,0x9c,0x5c,0xdc,0x3c,0xbc,0x7c
        0x02,0x82,0x42,0xc2,0x22,0xa2,0x62,0xe2,0x12,0x92,0x52,0xd2,0x32,0xb2,0x72
        0x0a,0x8a,0x4a,0xca,0x2a,0xaa,0x6a,0xea,0x1a,0x9a,0x5a,0xda,0x3a,0xba,0x7a
        0x06,0x86,0x46,0xc6,0x26,0xa6,0x66,0xe6,0x16,0x96,0x56,0xd6,0x36,0xb6,0x76
        0x0e,0x8e,0x4e,0xce,0x2e,0xae,0x6e,0xee,0x1e,0x9e,0x5e,0xde,0x3e,0xbe,0x7e
        0x01,0x81,0x41,0xc1,0x21,0xa1,0x61,0xe1,0x11,0x91,0x51,0xd1,0x31,0xb1,0x71
        0x09,0x89,0x49,0xc9,0x29,0xa9,0x69,0xe9,0x19,0x99,0x59,0xd9,0x39,0xb9,0x79
        0x05,0x85,0x45,0xc5,0x25,0xa5,0x65,0xe5,0x15,0x95,0x55,0xd5,0x35,0xb5,0x75
        0x0d,0x8d,0x4d,0xcd,0x2d,0xad,0x6d,0xed,0x1d,0x9d,0x5d,0xdd,0x3d,0xbd,0x7d
        0x03,0x83,0x43,0xc3,0x23,0xa3,0x63,0xe3,0x13,0x93,0x53,0xd3,0x33,0xb3,0x73
        0x0b,0x8b,0x4b,0xcb,0x2b,0xab,0x6b,0xeb,0x1b,0x9b,0x5b,0xdb,0x3b,0xbb,0x7b
        0x07,0x87,0x47,0xc7,0x27,0xa7,0x67,0xe7,0x17,0x97,0x57,0xd7,0x37,0xb7,0x77
        0x0f,0x8f,0x4f,0xcf,0x2f,0xaf,0x6f,0xef,0x1f,0x9f,0x5f,0xdf,0x3f,0xbf,0x7f
};

static unsigned char CSStab5[256]=
{
        0xff,0x7f,0xbf,0x3f,0xdf,0x5f,0x9f,0x1f,0xef,0x6f,0xaf,0x2f,0xcf,0x4f,0x8f
        0xf7,0x77,0xb7,0x37,0xd7,0x57,0x97,0x17,0xe7,0x67,0xa7,0x27,0xc7,0x47,0x87
        0xfb,0x7b,0xbb,0x3b,0xdb,0x5b,0x9b,0x1b,0xeb,0x6b,0xab,0x2b,0xcb,0x4b,0x8b
        0xf3,0x73,0xb3,0x33,0xd3,0x53,0x93,0x13,0xe3,0x63,0xa3,0x23,0xc3,0x43,0x83
        0xfd,0x7d,0xbd,0x3d,0xdd,0x5d,0x9d,0x1d,0xed,0x6d,0xad,0x2d,0xcd,0x4d,0x8d
        0xf5,0x75,0xb5,0x35,0xd5,0x55,0x95,0x15,0xe5,0x65,0xa5,0x25,0xc5,0x45,0x85
        0xf9,0x79,0xb9,0x39,0xd9,0x59,0x99,0x19,0xe9,0x69,0xa9,0x29,0xc9,0x49,0x89
        0xf1,0x71,0xb1,0x31,0xd1,0x51,0x91,0x11,0xe1,0x61,0xa1,0x21,0xc1,0x41,0x81
        0xfe,0x7e,0xbe,0x3e,0xde,0x5e,0x9e,0x1e,0xee,0x6e,0xae,0x2e,0xce,0x4e,0x8e
        0xf6,0x76,0xb6,0x36,0xd6,0x56,0x96,0x16,0xe6,0x66,0xa6,0x26,0xc6,0x46,0x86
        0xfa,0x7a,0xba,0x3a,0xda,0x5a,0x9a,0x1a,0xea,0x6a,0xaa,0x2a,0xca,0x4a,0x8a
        0xf2,0x72,0xb2,0x32,0xd2,0x52,0x92,0x12,0xe2,0x62,0xa2,0x22,0xc2,0x42,0x82
        0xfc,0x7c,0xbc,0x3c,0xdc,0x5c,0x9c,0x1c,0xec,0x6c,0xac,0x2c,0xcc,0x4c,0x8c
        0xf4,0x74,0xb4,0x34,0xd4,0x54,0x94,0x14,0xe4,0x64,0xa4,0x24,0xc4,0x44,0x84
        0xf8,0x78,0xb8,0x38,0xd8,0x58,0x98,0x18,0xe8,0x68,0xa8,0x28,0xc8,0x48,0x88
        0xf0,0x70,0xb0,0x30,0xd0,0x50,0x90,0x10,0xe0,0x60,0xa0,0x20,0xc0,0x40,0x80
};

static void CSSdescramble( unsigned char *key )
{
        unsigned int t1,t2,t3,t4,t5,t6;
        unsigned int i;

        t1= key[0] ^ 0x100;
        t2= key[1];
        t3=(*((unsigned int *)(key+2)));
        t4=t3&7;
        t3=t3*2+8-t4;
        t5=0;
          printf( "Keystate at start: %03x %02x %08x\n", t1, t2, t3 );
          printf( "output: " );
          for( i=0 ; i < 10 ; i++ )
          {
                t4=CSStab2[t2]^CSStab3[t1];
                t2=t1>>1;
                t1=((t1&1)<<8)^t4;
                t4=CSStab5[t4];
                t6=(((((((t3>>3)^t3)>>1)^t3)>>8)^t3)>>5)&0xff;
                t3=(t3<<8)|t6;
                t6=CSStab4[t6];
                t5+=t6+t4;
                printf( "%02x ",t5&0xff);
                t5>>=8;
          }
          printf( "\n" );
}


/*************************************************************
 *
 *   The Divide and conquer attack
 *
 *   Deviced and written by Frank A. Stevenson
 *
 *    ( frank@funcom.com )
 *   Released on a GPL license
 *
 *************************************************************/

static int RunLfsr2Backwards( int vStartState, int nSteps ) {
  unsigned int t1,t3,t6;
  int i,j;

  t3 = vStartState;
  for( i = 0 ; i < nSteps ; i++ ) {
    t1 = t3 & 0xff;
    t3 = ( t3 >> 8 );
    /* easy to code, and fast enough bruteforce search for byte shifted in */
    for( j=0 ; j < 256 ; j++ ) {
      t3 = (t3 & 0x1ffff) | ( j << 17 );
      t6=(((((((t3>>3)^t3)>>1)^t3)>>8)^t3)>>5)&0xff;
      if( t6 == t1 ) break;
    }
  }
  return t3;
}

static unsigned char invtab4[256];

static void CSScracker( unsigned char* pStream, unsigned char *pTableA, unsigned c
  unsigned int t1,t2,t3,t4,t5,t6;
  unsigned int nTry;
unsigned int vCandidate;
int i;

/* Test that pTableA is a permutation */
memset( invtab4, 0, 256 );
for( i = 0 ; i < 256 ; i++ ) invtab4[ pTableA[i] ] = 1;
for( i = 0 ; i < 256 ; i++ ) if( invtab4[ i ] != 1 ) {
  printf( "Permutation error\n" );
  exit( -1 );
}

/* initialize the inverse of table4 */
for( i = 0 ; i < 256 ; i++ ) invtab4[ pTableA[i] ] = i;

for( nTry = 0 ; nTry < 65536 ; nTry++ ) {
  t1 = nTry >> 8 | 0x100;
  t2 = nTry & 0xff;
  t3 = 0;
  t5 = 0;

  /* iterate cipher 3 times to reconstruct LFSR2 16/17 bits */
  for( i = 0 ; i < 3 ; i++ ) {
    /* advance LFSR1 normaly */
    t4=CSStab2[t2]^CSStab3[t1];
    t2=t1>>1;
    t1=((t1&1)<<8)^t4;
    t4=pTableB[t4];
    /* deduce t6 & t5 */
    t6 = pStream[ i ];
    if( t5 ) t6 = ( t6 + 0xff )&0x0ff;
    if( t6 < t4 ) t6 += 0x100;
    t6 -= t4;
    t5 += t6 + t4;
    t6 = invtab4[ t6 ];
    /* printf( "%02x/%02x ", t4, t6 ); */
    /* feed / advance t3 / t5 */
    t3 = (t3 << 8) | t6;
    t5 >>= 8;
  }

  /* Guess the most significant bit of LFSR2 */
  vCandidate =  RunLfsr2Backwards( t3 , 3 );
  if( ( vCandidate & 0x08 ) == 0 ) {
    t3 |=  0x01000000;
    vCandidate =  RunLfsr2Backwards( t3 , 3 );
  }
  if( ( vCandidate & 0x08 ) == 0 ) {
    printf( "Failed to guess bit - exiting\n" );
    exit( -1 );
  }

  /* iterate 2 more times to validate candidate key */
  for( ; i < 5 ; i++ ) {
    t4=CSStab2[t2]^CSStab3[t1];
    t2=t1>>1;
    t1=((t1&1)<<8)^t4;
    t4=pTableB[t4];
    t6=(((((((t3>>3)^t3)>>1)^t3)>>8)^t3)>>5)&0xff;
    t3=(t3<<8)|t6;
    t6=pTableA[t6];
    t5+=t6+t4;
    if( (t5 & 0xff) != pStream[i] ) break;
    t5>>=8;
```

```
      }

      if( i == 5 ) {
        /* Key was found - print out result */
        t4 = ( vCandidate / 2 ) & 0xfffff8;
        t4 |= vCandidate & 0x7;
        /* printf( "Candidate: %03x %02x %08x\n", 0x100|(nTry>>8),nTry&0x0ff, vCandi
        printf( "  Possible Player key %02x %02x %02x %02x %02x\n", nTry>>8, nTry&0x
      }
    }
  }

  /* simple function to convert hex bytes to int */
  /* note: will give random results if nonhex digits are input */

  static char hexdigits[17] = "0123456789abcdef\0";

  static int HexByteToInt( const char *pNumber ) {
   char ch;
   int r;

   ch = tolower( pNumber[0] );
   r = 16 * (int)( strchr( hexdigits, ch ) - hexdigits );
   ch = tolower( pNumber[1] );
   r+= (int)( strchr( hexdigits, ch ) - hexdigits );

   return r & 0x0ff;  /* invalid input will have produce garbage */
  }


 /* Revert mangling function - and crack keys*/
 static int unmangle ( unsigned char* in , unsigned char *out ) {
   unsigned char A[5];
   unsigned char B[5];
   unsigned char C[5];
   unsigned char k[6];
   int i,j;

   /* Recover mangling key */
   memcpy( A, in, 5 );
   memcpy( C, out, 5 );
   k[5] = 0;

   for( i=0 ; i < 256 ; i++ ) {
     k[4] = i;
     for( j = 4 ; j >= 2 ; j-- ) {
       B[j] = k[j] ^ CSStab1[ A[j] ] ^ A[j-1];
       B[j-1] = CSStab1[ B[j] ] ^ k[j] ^ C[j];
       k[j-1] = A[j-2] ^ CSStab1[ A[j-1] ] ^ B[j-1];
     }
     B[0] = CSStab1[ B[1] ] ^ k[1] ^ C[1];
     k[0] = B[0] ^ CSStab1[ A[0] ] ^ B[4];

     if( ( CSStab1[ B[0] ] ^ k[0]  )== C[0] ) {
       printf( "Possible mangling key: %02x %02x %02x %02x %02x\n", k[0], k[1], k[2
       CSScracker( k, CSStab4, CSStab4 );

     }
 }
```

```
    return 0;
 }

/* Main function */
int main( int argc, char* argv[] ){
  int i;
  unsigned char in[5] = { 0,0,0,0,0 };
  unsigned char out[5] = { 0,0,0,0,0 };

  if( argc != 11 ) {
    printf( "Usage: %s xx xx xx xx xx yy yy yy yy yy ( Disc key / Encrypted Disk k
    return -1;
  }

  for( i = 0; i < 5 ; i++ ) {
    in[ i ] = HexByteToInt( argv[i+1] );
    out[ i ] = HexByteToInt( argv[i+6] );
  }

  /* search for key */
  printf( "Recovering Key\n");
  unmangle( in, out );

  return( 0 );
}
```